Java Chapter 10: Files

Contents

Time required: 60 minutes

DRY

Don't **R**epeat **Y**ourself

Files

Most of the programs we have seen so far are transient in the sense that they run for a short time and produce some output. When they end, their data disappears. If you run the program again, it starts with a clean slate.

Other programs are persistent: they run for a long time (or all the time); they keep at least some of their data in permanent storage (a hard drive, for example); and if they shut down and restart, they pick up where they left off.

Examples of persistent programs are operating systems, which run pretty much whenever a computer is on, and web servers, which run all the time, waiting for requests to come in on the network.

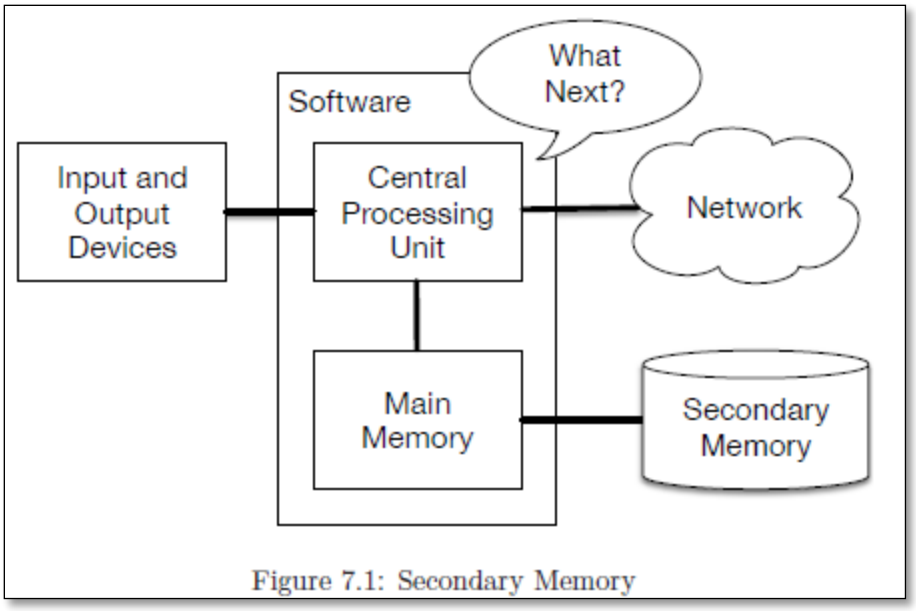


Figure 7.1: Secondary Memory

Secondary memory is not erased when the power is turned off. Or in the case of a USB flash drive, the data we write from our programs can be removed from the system and transported to another system.

Tutorial 8.1 – Reading a Text File

In the same folder, create a txt file named **example.txt** Include the following text.

```
Hello
This is a sample text file.
Bye!
```

Create and test the program as listed. Name it **file_read.py**

```python
"""
    Name: file_read.py
    Author:
    Created:
    Purpose: Read and display a text file
"""


def main():

    # Open a file in the same folder as the program
    # We create an object named text_file
    # This is called a file handle
    text_file = open("example.txt", "r")

    # Read the entire contents of the file into a string
    file_contents = text_file.read()

    # Close the file handle
    text_file.close()

    # Print the string
    print(file_contents)


# If a standalone program, call the main function
# Else, use as a module
if __name__ == "__main__":
    main()
```

Tutorial 8.2 – Writing a Text File

Create and test the following Python program called **file_rockstars_write.py**

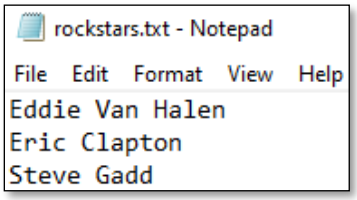
This program uses try: except: to provide error handling. If there was an exception, inform the user.

Let the user know the operation was successful by including a print statement in the try block.

```python
"""
    Name: file_rockstars_write.py
    Author:
    Created:
    Purpose: Write 3 lines of data to a text file
"""


def main():

    # Catch any exceptions
    try:
        # Create a file handle
        # for writing to rockstars.txt
        with open("rockstars.txt", "w") as rock_file:

            # Write the names of three rock stars to the file
            # Substitute your favorites
            # \n is an escape character creating a new line
            rock_file.write(f"Eddie Van Halen\n")
            rock_file.write(f"Eric Clapton\n")
            rock_file.write(f"Steve Gadd\n")
            print(f"File written successfully.")

    # Let the user know there was an exception
    except:
        print(f"The file was not written.")


# Call the main function
if __name__ == "__main__":
    main()
```

Open **rockstars.txt** to check your work.

Example run:



Tutorial 8.3 – Reading a Text File Line by Line

Create a new program called **file_rockstars_read.py** This program reads each line into a separate string.

```
"""
    Name: file_rockstars_read.py
    Author:
    Created:
    Purpose: Read and display a text file line by line
"""


def main():

    # Catch any exceptions in the program
    try:
        # Open a file in the same folder as the program
        with open('rockstars.txt', 'r') as text_file:

            # Read each line into a separate string
            line1 = text_file.readline()
            line2 = text_file.readline()
            line3 = text_file.readline()

        # Print the strings
        print(line1)
        print(line2)
        print(line3)
        print('The file was successfully read.')

    # Let the user know if there was an exception
    except:
        print('There was a problem reading the file.')


# Call the main function
if __name__ == '__main__':
    main()
```

Because of the \n newline character we added when we wrote the file, there is a space between each line when we print out the strings.

```
Eddie Van Halen

Eric Clapton

Steve Gadd

The file was successfully read.
```

```python
"""
    Name: file_rockstars_read_rstrip.py
    Author:
    Created:
    Purpose: Read and display a text file line by line
"""


def main():

    # Catch any exceptions in the program
    try:
        # Open a file in the same folder as the program
        with open('rockstars.txt', 'r') as text_file:

            # Read each line into a separate string
            line1 = text_file.readline()
            line2 = text_file.readline()
            line3 = text_file.readline()

        # Strip \n from each string with the rstrip function
        line1 = line1.rstrip('\n')
        line2 = line2.rstrip('\n')
        line3 = line3.rstrip('\n')

        # Print the strings
        print(line1)
        print(line2)
        print(line3)

        print('The file was successfully read.')

    # Let the user know if there was an exception
    except:
        print('There was a problem reading the file.')


# Call the main function
if __name__ == '__main__':
    main()
```

The text file displayed without \n extra new line characters.

```
Eddie Van Halen
Eric Clapton
Steve Gadd
The file was successfully read.
```

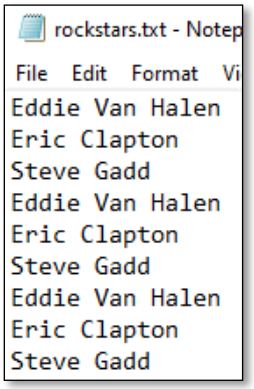
Tutorial 8.4 – Appending to a Text File

'a'	Opens a file for appending at the end of the file without erasing the contents. Creates a new file if it does not exist.

Open **file_rockstars_write.py**. The only change needed is to change the 'w' to an 'a'. Run the program a couple times. Open the text file to make sure the program worked.

```
"""
    Name: file_rockstars_append.py
    Author:
    Created:
    Purpose: Append 3 lines of data to a text file
"""


def main():

    # Catch any exceptions
    try:
        # Open a file for appending in the same folder named rockstars.txt
        with open('rockstars.txt', 'a') as rock_file:

            # Append the names of three rock stars to the file
            # Substitute your favorites
            # \n is an escape character creating a new line
            rock_file.write('Eddie Van Halen' + '\n')
            rock_file.write('Eric Clapton' + '\n')
            rock_file.write('Stevie Nicks' + '\n')

            print('File written successfully')

    # Let the user know there was an exception
    except:
        print('Something went wrong and caused an exception')


# Call the main function.
main()
```

Example run:

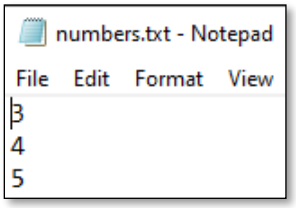


Tutorial 8.5 – Writing Numbers to a Text File

Numbers must be converted to a string before they can be written to a text file. We will use Fstrings to convert the numbers to strings.

```
"""
    Name: file_numbers_write.py
    Author:
    Created:
    Purpose: Numbers must be converted to strings before they
             are written to a text file.
"""
# Constant for filename
FILE_NAME = 'numbers.txt'


def main():

    # Catch any exceptions
    try:
        # Open a file for writing
        with open(FILE_NAME, 'w') as number_file:

            # Get three numbers from the user.
            number1 = int(input('Enter a whole number: '))
            number2 = int(input('Enter another whole number: '))
            number3 = int(input('Enter another whole number: '))

            # Write the numbers to the file using Fstrings
            number_file.write(f'{number1}\n')
            number_file.write(f'{number2}\n')
            number_file.write(f'{number3}\n')

            # Let the user know it worked
            print('Data was written to numbers.txt')

    # Let the user know there was trouble
    except:
        print('There was trouble writing to the file.')


# Call the main function.
if __name__ == '__main__':
    main()
```

Example run:



Tutorial 8.6 – Read Numbers from a Text File

Numbers stored as text in a text file must be converted from string to numbers before they can be used in calculations.

```
"""
    Name: file_read_numbers.py
    Author:
    Created:
    Purpose: Numbers must be converted from strings to ints
"""
# Constant for filename
FILE_NAME = 'numbers.txt'


def main():

    # Catch any exceptions
    try:
        # Open a file for reading
        with open(FILE_NAME, 'r') as number_file:

            # Read 3 numbers from a file
            number1 = int(number_file.readline())
            number2 = int(number_file.readline())
            number3 = int(number_file.readline())

        # Sum the numbers
        total = number1 + number2 + number3

        # Display the numbers and the total
        # A different way of printing numbers as strings
        print(f'The numbers are: {number1} {number2} {number3}')
        print(f'The total is: {total}')

    # Let the user know there was trouble
    except:
        print('There was trouble reading the file.')


# Call the main function.
if __name__ == '__main__':
    main()
```

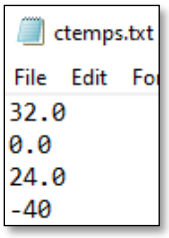
Example run:

```
The numbers are: 3 4 5
The total is: 12
```

Tutorial 8.7 – Reading and Writing to a Text File

We will write a program that reads a list of temperatures from a file called **ctemps.txt**, converts those temperatures to Fahrenheit, and writes the results to a file called **ftemps.txt**.

Create a text file named **ctemps.txt** in the current folder and put the following numbers in it. Don't put any extra line returns in it, stop at typing at -40.

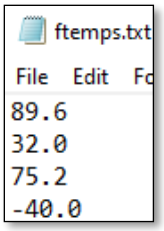


Create and test the following program named **convert_temperatures.py**

```python
'''
    Name: convert_temperatures.py
    Author:
    Created:
    Purpose: Read a list of temperatures from a file, convert them to Fahrenheit
    Write the results to a file and to the screen
'''
# Constant for filename
CFILE = 'ctemps.txt'
FFILE = 'ftemps.txt'

def main():

    # Catch any exceptions
    try:
        # Open a file for writing
        cfile = open( CFILE, 'r')
        ffile = open( FFILE, 'w')

        # Strip the newline character off of every line in the file
        ctemp = [line.strip() for line in cfile ]

        # Loop until all the temperatures have been converted
        for t in ctemp:
            # Convert the temperature to Fahrenheit
            ftemp = float(t) * (9.0 / 5.0) + 32.0
            # Write the temperature to the file
            ffile.write( f'{ftemp}\n' )
            # Print the conversion to the screen
            print(f'{t} Celcius equals {ftemp} Fahrenheit')

    # Let the user know there was an exception
    except:
        print('A file exception occured.')

    # Regardless of what happens, the file is closed
    finally:
        # Close both files
        cfile.close()
        ffile.close()

# Call the main function
if __name__ == '__main__':
    main()
```

Example run:

```
32.0 Celcius equals 89.6 Fahrenheit
0.0 Celcius equals 32.0 Fahrenheit
24.0 Celcius equals 75.2 Fahrenheit
-40 Celcius equals -40.0 Fahrenheit
```



Assignment Submission

1. Attach the pseudocode.
2. Attach the program files.
3. Attach screenshots showing the successful operation of the program.
4. Submit in Blackboard.